Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2012, Article ID 872619, 2 pages
doi:10.1155/2012/872619

Editorial

Software Quality Assurance Methodologies and Techniques

Chin-Yu Huang,[1] Hareton Leung,[2] Wu-Hon Francis Leung,[3] and Osamu Mizuno[4]

[1] *Department of Computer Science and Institute of Information Systems and Applications, National Tsing Hua University, Hsinchu 30013, Taiwan*
[2] *Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong*
[3] *Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616, USA*
[4] *Graduate School of Science and Technology, Kyoto Institute of Technology, Kyoto 606-8585, Japan*

Correspondence should be addressed to Chin-Yu Huang, cyhuang@cs.nthu.edu.tw

Received 18 July 2012; Accepted 18 July 2012

Software quality assurance (SQA) is a planned and systematic pattern of actions necessary to provide adequate confidence that a software product conforms to requirements during software development. SQA consists of methodologies and techniques of assessing the software development processes and methods, tools, and technologies used to ensure the quality of the developed software. SQA is typically achieved through the use of well-defined standard practices, including tools and processes, for quality control to ensure the integrity and reliability of software. This special issue presents new research works along these directions, and we received 21 submissions and accepted five of them after a thorough peer-review process. The acceptance rate of this special issue is around 24%. The resultant collection provides a number of useful results. These accepted papers cover a broad range of topics in the research field of SQA, including software validation, verification, and testing, SQA modeling, certification, evaluation, and improvement, SQA standards and models, SQA case studies, data analysis and risk management.

For example, in "*Specifying process views for a measurement, evaluation, and improvement strategy,*" P. Becker, P. Lew, and L. Olsina developed a specific strategy called SIQinU (strategy for understanding and improving quality in use), which recognizes problems of quality in use through evaluation of a real system-in-use situation and proposes product improvements by understanding and making changes to the product's attributes. They used UML 2.0 activity diagrams and the SPEM profile to stress the functional, informational, organizational, and behavioral views for the SIQinU process.

In the paper "*Program spectra analysis with theory of evidence,*" R. Hewett proposed a spectrum-based approach to fault localization using the Dempster-Shaffer theory of evidence. Using mathematical theories of evidence for uncertainty reasoning, the proposed approach estimates the likelihood of faulty locations based on evidence from program spectra. Evaluation results show that their approach is at least as effective as others with an average effectiveness of 85.6% over 119 versions of the programs.

In the paper entitled "*An empirical study on the impact of duplicate code,*" K. Hotta et al. presented an empirical study on the impact of the presence of duplicate code on software evolution. They assumed that if duplicate code is modified more frequently than nonduplicate code, the presence of duplicate code affects software evolution, and compared the stability of duplicate code and non-duplicate code. They conducted an experiment on 15 open-source software systems, and the result showed that duplicate code was less frequently modified than nonduplicate code and, in some cases, duplicate code was intensively modified in a short period though duplicate code was more stable than nonduplicate code in the whole development period.

The next paper by X. Xiao and T. Dohi, "*A comparative study of data transformations for wavelet shrinkage estimation with application to software reliability assessment,*" applied the wavelet-based techniques to estimate the software intensity function. Some data transformations were employed to preprocess the software-fault count data. Throughout the numerical evaluation, the authors concluded that the

wavelet-based estimation methods have much more potential applicability than the other data transformations to the software reliability assessment.

In the last paper "*Can faulty modules be predicted by warning messages of static code analyzer?*," O. Mizuno and M. Nakai proposed a detection method of fault-prone modules based on the spam filtering technique—fault-prone filtering. For the analysis, the authors tried to state two questions: "can fault-prone modules be predicted by applying a text filter to the warning messages of static code analyzer?" and "is the performance of the fault-prone filtering becomes better with the warning messages of a static code analyzer?". The results of experiments show that the answer to the first question is "yes." But for the second question, the authors found that the recall becomes better than the original approach.

In summary, this special issue serves as a platform for researchers and practitioners to present theory, results, experience, and other advances in SQA. Hopefully, you will enjoy this publication, and we look forward to your feedback and comments.

Chin-Yu Huang
Hareton Leung
Wu-Hon Francis Leung
Osamu Mizuno